Project 2

Daniel Fredin, Junhan Li, & Eric Chen

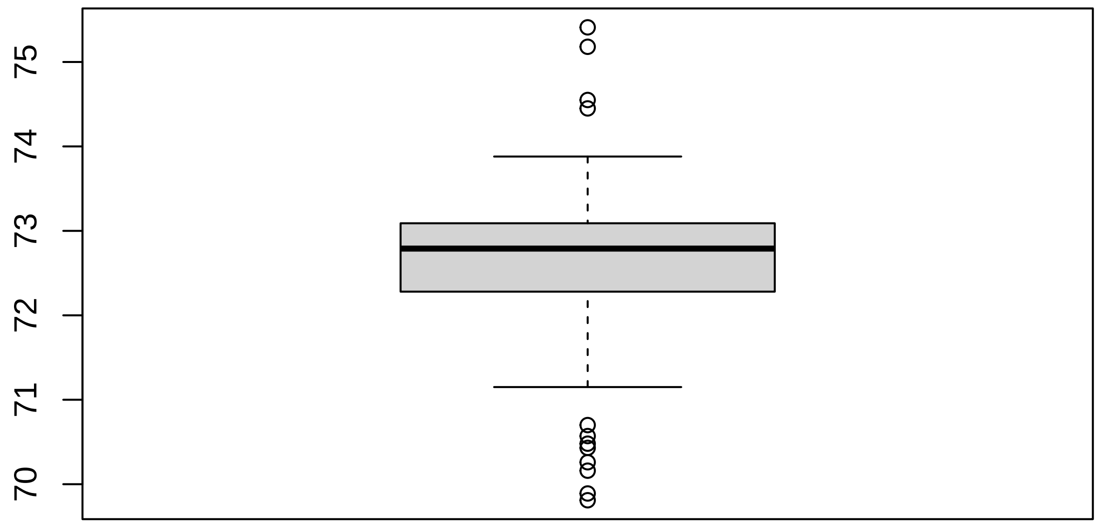
Problem 1

```r
glass <- read.csv("glass.csv")
```

Initial assumptions - Dependent variable should be approximately normally distributed - Variables should related to each other linear - use a matrix scatterplot - There should be no significant outliers - Kept at minimum

Test 1 Remove some outliers

```r
boxplot(glass$Si)
```



```r
outliers <- boxplot.stats(glass$Si)$out
max(outliers)
```

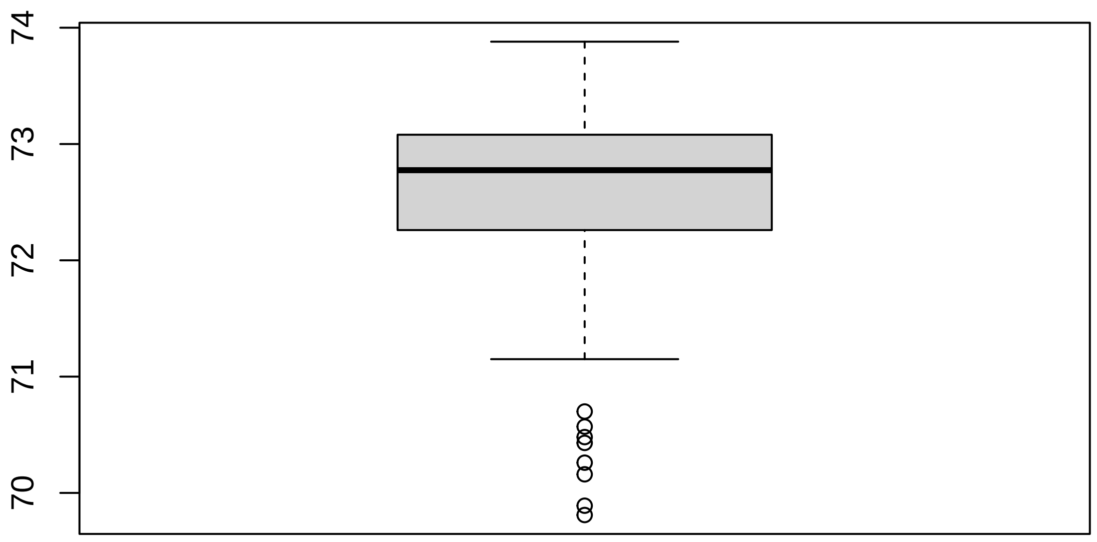
```
## [1] 75.41
```

```r
min(outliers)
```

```
## [1] 69.81
```

```r
glass <- subset(glass, glass$Si<74)
glass <- subset(glass, glass$Si != max(outliers))

boxplot(glass$Si)
```

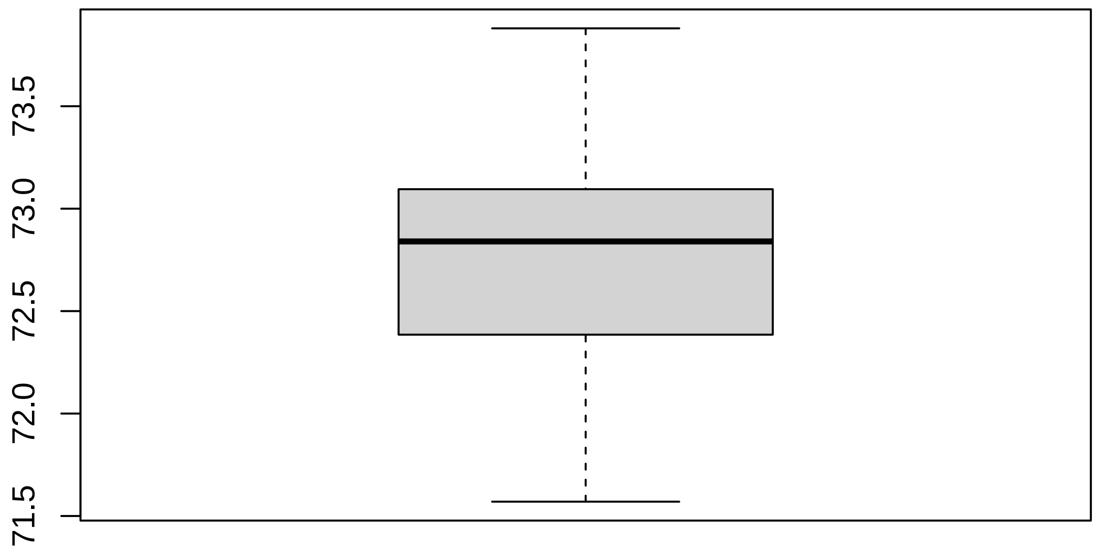


```r
max(boxplot.stats(glass$Si)$out)
```

```
## [1] 70.7
```

Test 2 Remove all outliers(Currently our best option)

```r
glass <- subset(glass, glass$Si<74)
glass <- subset(glass, glass$Si>71.5)
boxplot(glass$Si)
```



Test 3: Use matrix scatterplot to show that the variables related to each others linearly

- response variable will be (Si) # Part (a)
- Select four variables of my choice
- use visualization plot to tell a story, making use the variables selected
- Using color,faceting, theme in ggplot2

```r
# Show values of
```

```r
# We will first check for the normality of dependent variable
shapiro.test(glass$Si)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  glass$Si
## W = 0.97376, p-value = 0.001015
```

- 2 Paragraphs
  - why we choose variables
  - what you see from graph

Part (b) Fit at least two models(should be same type?)

We will use MLR

We will first check for the normality of dependent variable

```
# We will first check for the normality of dependent variable
shapiro.test(glass$Si)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glass$Si
## W = 0.97376, p-value = 0.001015
```

Since the shapiro test gives us a p-value 2.175e-09, which is less than 0.05, we will need to take the square root of our dependent variable to make it normal.

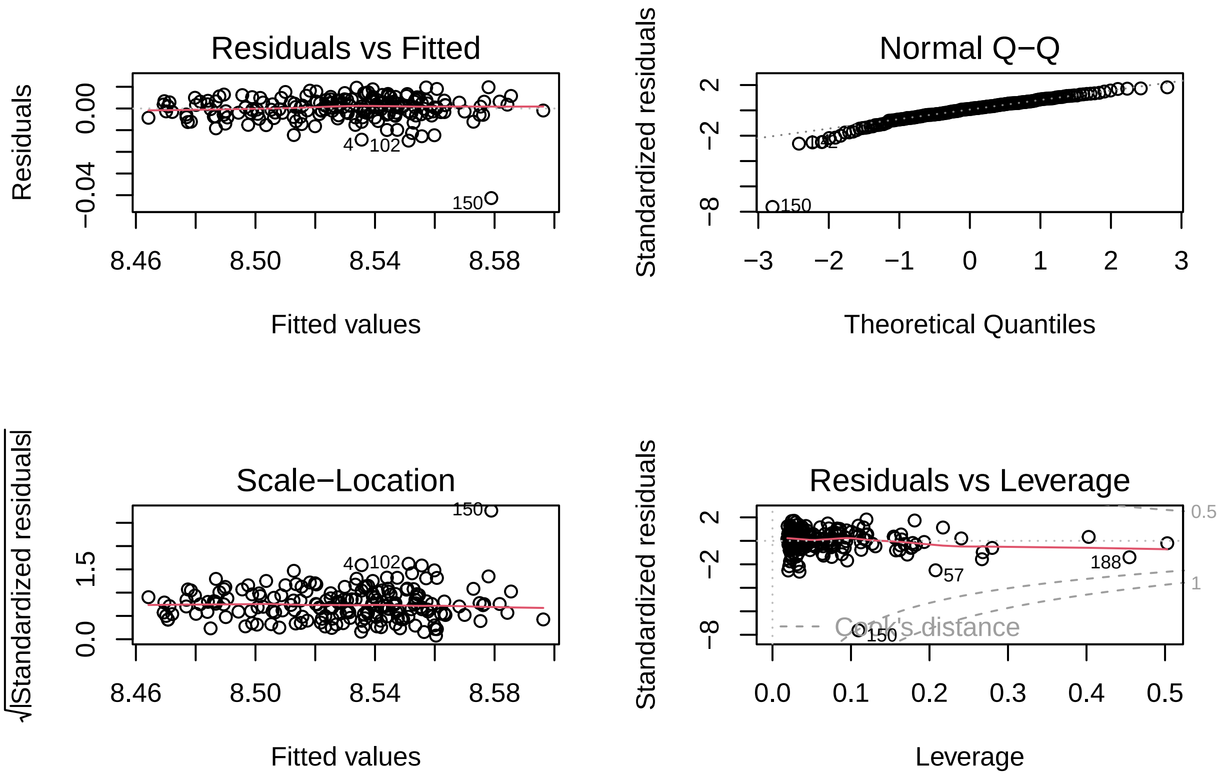
Factor data directly

```
# We will first check for the normality of dependent variable

glass_model1 <- lm(sqrt(Si) ~., data = glass)
shapiro.test(residuals(glass_model1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(glass_model1)
## W = 0.85589, p-value = 1.285e-12
```

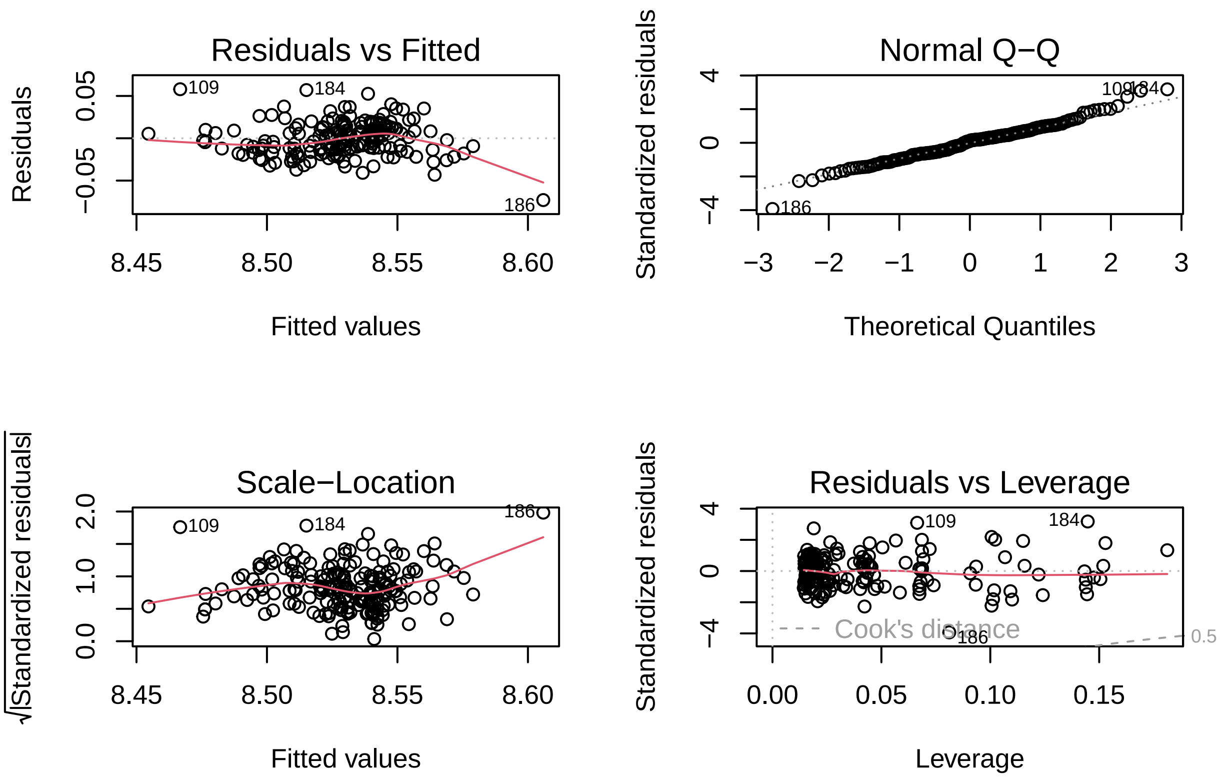
```
par(mfrow = c(2,2))
plot(glass_model1)
```



```
glass_model2 <- lm(sqrt(Si)  ~ RI + factor(type) + Na  , data = glass)
shapiro.test(residuals(glass_model2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(glass_model2)
## W = 0.98701, p-value = 0.0708
```

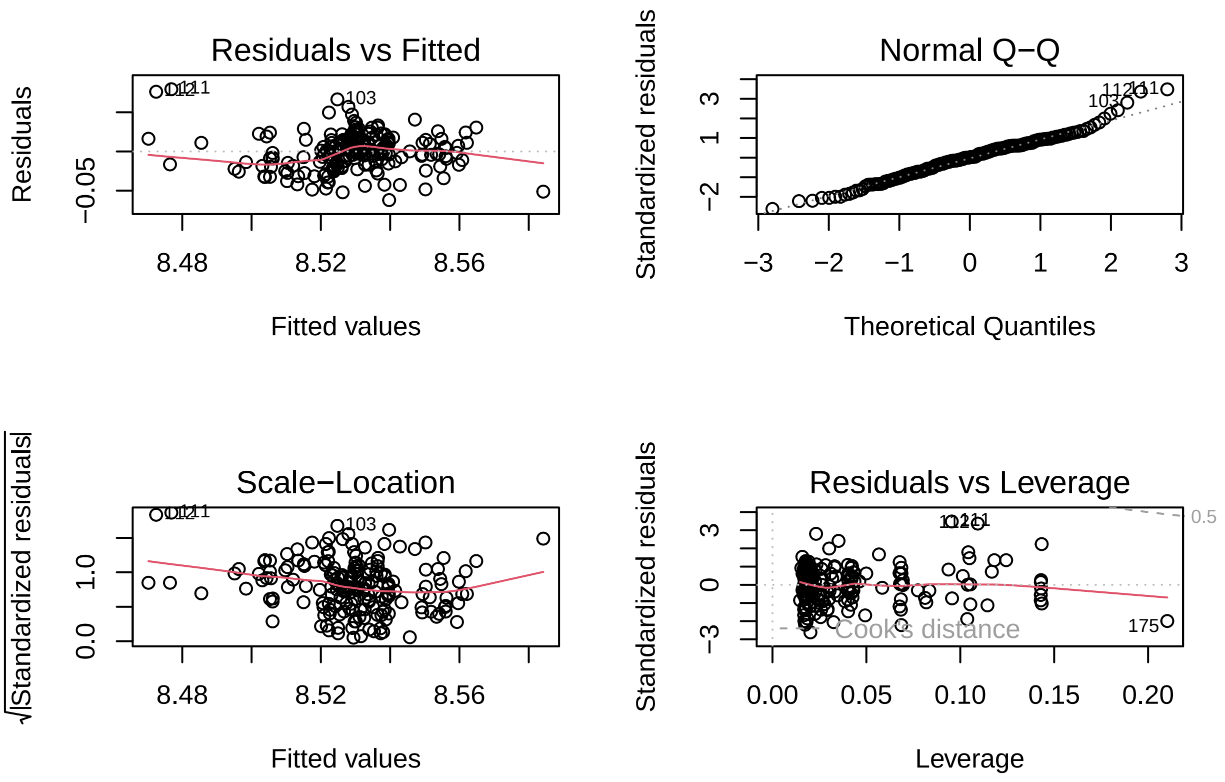
```
par(mfrow = c(2,2))
plot(glass_model2)
```



```
glass_model3 <- lm(sqrt(Si)  ~ RI + factor(type) + Fe  , data = glass)
shapiro.test(residuals(glass_model3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(glass_model3)
## W = 0.98585, p-value = 0.04783
```

```
par(mfrow = c(2,2))
plot(glass_model3)
```



After we have take the square root of the dpendent variable Si, we can

- select the best competing model for statistical analyses

```
summary(glass_model1)
```

```
##
## Call:
## lm(formula = sqrt(Si) ~ ., data = glass)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.041330 -0.002430  0.000623  0.003239  0.009801
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.0272928  0.0335245 299.103  < 2e-16 ***
## RI          -0.0003628  0.0004436  -0.818   0.4145
## Na          -0.0554272  0.0011700 -47.372  < 2e-16 ***
## Mg          -0.0550156  0.0014273 -38.545  < 2e-16 ***
## Al          -0.0562445  0.0019249 -29.220  < 2e-16 ***
## K           -0.0594947  0.0025889 -22.981  < 2e-16 ***
## Ca          -0.0548731  0.0017215 -31.875  < 2e-16 ***
## Ba          -0.0548371  0.0025319 -21.658  < 2e-16 ***
## Fe          -0.0181665  0.0044480  -4.084 6.63e-05 ***
## typeHead     0.0062513  0.0035947   1.739   0.0837 .
## typeTabl     0.0024992  0.0032416   0.771   0.4417
## typeVeh     -0.0034086  0.0029639  -1.150   0.2517
## typeWinF     0.0008376  0.0026120   0.321   0.7488
## typeWinNF   -0.0023427  0.0024064  -0.974   0.3316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005745 on 181 degrees of freedom
## Multiple R-squared:  0.9631, Adjusted R-squared:  0.9605
## F-statistic: 363.5 on 13 and 181 DF,  p-value: < 2.2e-16
```

```
summary(glass_model2)
```

```
##
## Call:
## lm(formula = sqrt(Si) ~ RI + factor(type) + Na, data = glass)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07301 -0.01223  0.00234  0.01135  0.05792
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.8906998  0.0340254 261.296  < 2e-16 ***
## RI                -0.0060524  0.0006171  -9.807  < 2e-16 ***
## factor(type)Head   0.0351995  0.0089425   3.936 0.000117 ***
## factor(type)Tabl   0.0180661  0.0104816   1.724 0.086434 .
## factor(type)Veh   -0.0262537  0.0082890  -3.167 0.001797 **
## factor(type)WinF  -0.0152223  0.0067998  -2.239 0.026359 *
## factor(type)WinNF -0.0184177  0.0068267  -2.698 0.007616 **
## Na                -0.0263335  0.0026343  -9.996  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01937 on 187 degrees of freedom
## Multiple R-squared:  0.5668, Adjusted R-squared:  0.5505
## F-statistic: 34.95 on 7 and 187 DF,  p-value: < 2.2e-16
```

```
summary(glass_model3)
```

```
##
## Call:
## lm(formula = sqrt(Si) ~ RI + factor(type) + Fe, data = glass)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.062094 -0.016115  0.000469  0.014528  0.079437
##
## Coefficients:
##                     Estimate Std. Error  t value Pr(>|t|)
## (Intercept)        8.5570226  0.0079440 1077.166  < 2e-16 ***
## RI                -0.0058449  0.0007638   -7.653 1.02e-12 ***
## factor(type)Head  -0.0119294  0.0095098   -1.254 0.211248
## factor(type)Tabl  -0.0248065  0.0119603   -2.074 0.039443 *
## factor(type)Veh   -0.0431569  0.0100782   -4.282 2.95e-05 ***
## factor(type)WinF  -0.0291836  0.0082514   -3.537 0.000511 ***
## factor(type)WinNF -0.0296748  0.0083371   -3.559 0.000471 ***
## Fe                -0.0070476  0.0184762   -0.381 0.703310
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02399 on 187 degrees of freedom
## Multiple R-squared:  0.3358, Adjusted R-squared:  0.3109
## F-statistic:  13.5 on 7 and 187 DF,  p-value: 4.473e-14
```

```
#. AIC

compareLM(glass_model1, glass_model2, glass_model3)
```

```
## $Models
##   Formula
## 1 "sqrt(Si) ~ RI + Na + Mg + Al + K + Ca + Ba + Fe + type"
## 2 "sqrt(Si) ~ RI + factor(type) + Na"
## 3 "sqrt(Si) ~ RI + factor(type) + Fe"
##
## $Fit.criteria
##   Rank Df.res     AIC    AICc     BIC R.squared Adj.R.sq    p.value Shapiro.W
## 1   14    181 -1443.0 -1441.0 -1394.0    0.9631   0.9605 3.994e-122    0.8559
## 2    8    187  -974.9  -974.0  -945.5    0.5668   0.5505  7.120e-31    0.9870
## 3    8    187  -891.6  -890.6  -862.1    0.3358   0.3109  4.473e-14    0.9859
##   Shapiro.p
## 1 1.285e-12
## 2 7.080e-02
## 3 4.783e-02
```

- Argue your choice
- Perform optimization in R, compute the regression coefficients

The lower the value for AIC, the better the fit of the model. The absolute value of the AIC value is not important. It can be positive or negative.

Preconlcusion: Model 2 is a lost better.

Part (c) Use result of best model, answer following questions

(i): Explain coefficient of determination (ii): Find Least-squares estimates: Basically Regression - (iii): Interpret slopes and intercept of the problem - determine the significant predictors at 10% level of significance (iv): Perform Residual analysis - the four tests - four plots group together

Par(d): separate observations into

- group 1: type = WinF
- group 2: type= WonNF Variable A1, level alpha = 0.1

Problem 3

a)

ii) 10-fold Cross-Validation

Model Accuracy

```r
# Predict outcome using model from training data based on testing data
predictions <- predict(cvmodel, newdata=test)

# Create confusion matrix to assess model fit/performance on test data
con_matx <- confusionMatrix(data=predictions, test$COMMIT)
con_matx
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  13   4
##        yes  2  11
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007155
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : 0.6830914
##
##             Sensitivity : 0.8667
##             Specificity : 0.7333
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8462
##              Prevalence : 0.5000
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : no
##
```

b)

Our objective is to optimize the cost function in order to identify the most economical approach for fulfilling the Christmas order. This entails striving for the lowest cost per day to achieve maximum cost efficiency. By minimizing expenses at each factory, we can ensure a cost-effective process for meeting the demands of the Christmas order.

```r
factories <- c("Factory A", "Factory B", "Factory C")
toys <- c("Cars", "Animals", "Robots")

# Define the coefficients of the objective function
costs <- c(1000, 2100, 1500)

# Define the constraint matrix
constraint_matrix <- matrix(c(30, 20, 30,
                              40, 50, 10,
                              50, 40, 15), nrow = 3, byrow = TRUE)

# Define the right-hand side of the constraints
constraint_limits <- c(5000, 3000, 2500)

# Set the direction of optimization (minimization)
constraint_directions <- c(">=", ">=", ">=")
direction <- "min"

# Solve the linear programming problem
min_solution <- lp(direction = direction,
               objective.in = costs,
               const.mat = constraint_matrix,
               const.dir = constraint_directions,
               const.rhs = constraint_limits,
               compute.sens=TRUE)


# Check if a solution was found
if (min_solution$status == 0) {
  # Print the optimal solution
  cat("The optimal solution is:\n")
  cat("Factory A:",min_solution$solution[1], "days","\n")
  cat("Factory B:",min_solution$solution[2], "days","\n")
  cat("Factory C:",min_solution$solution[3], "days","\n\n")
  # Print the minimum cost
  cat("The value of the objective function at the optimal solution is:\n")
  cat("Minimum cost: $",min_solution$objval)
} else {
  # No feasible solution found
  print("No feasible solution found.")
}
```

```
## The optimal solution is:
## Factory A: 166.6667 days
## Factory B: 0 days
## Factory C: 0 days
##
## The value of the objective function at the optimal solution is:
## Minimum cost: $ 166666.7
```

The most efficient approach entails running Factory A for approximately 166.67 days, resulting in a minimal cost of $166,666.70, whereas Factory B and Factory C remain non-operational. By adopting this strategy, the company can achieve the most cost-effective outcome.

When considering the optimal solution, it is important to analyze the cost implications of running the factories for different durations. In this scenario, operating Factory for approximately 166.67 days leads to the minimum overall cost. By halting the operations of Factory B and Factory C, the company can avoid additional expenses associated with their functioning.

```r
# Create an empty data frame to store the results
result_df <- data.frame()
factories <- c("Factory A", "Factory B", "Factory C")
toys <- c("Cars", "Animals", "Robots")



# Iterate over the range of 1:3
for (i in 1:3) {

  # Store values for rows and columns in the data frame
  row_df <- data.frame(Factory = factories[i],
                       min_days = min_solution$solution[i],
                       min_cost = min_solution$solution[i]*costs[i])


  result_df <- rbind(result_df, row_df)

}

colnames(result_df) <- c("",
                         "Minimum # of days",
                         "Minimum costs")

# Print the resulting data frame
cat("The optimal solution is:")
```

```
## The optimal solution is:
```

```r
print(result_df)
```

```
##             Minimum # of days Minimum costs
## 1 Factory A          166.6667      166666.7
## 2 Factory B            0.0000           0.0
## 3 Factory C            0.0000           0.0
```

```r
# Sensitivity Analysis

min_solution$sens.coef.from
```

```
## [1]    0.0000  666.6667 1000.0000
```

```r
min_solution$sens.coef.to
```

```
## [1] 1.5e+03 1.0e+30 1.0e+30
```

c)

```
compute_weibull_stats <- function(shape, scale) {
  # Compute the mean
  mean_value <- scale * gamma(1 + (1/shape))


  # # Compute the median
  # median_value <- scale * qweibull(0.5, shape, scale)


  # Compute the median
  median_value <- scale * (log(2)^(1/shape))


  # Compute the mode
  if (shape > 1)
    mode_value <- scale * ((shape - 1) / shape)^(1/shape)
  else
    mode_value <- 0

  # Compute the variance
  variance_value <- scale^2 * (gamma(1 + (2/shape)) - (gamma(1 + (1/shape)))^2)

  # Return the computed statistics as a named list
  return(list(mean = mean_value,
              median = median_value,
              mode = mode_value,
              variance = variance_value))
}

# lambda = scale
# K = shape

# Example usage
scale_param <- 6
shape_param <- 1

stats <- compute_weibull_stats(shape_param, scale_param)

# Accessing the computed statistics
mean_value <- stats$mean
median_value <- stats$median
mode_value <- stats$mode
variance_value <- stats$variance

# Printing the computed statistics
cat("Mean:", mean_value, "\n")
```

```
## Mean: 6
```

```
cat("Median:", median_value, "\n")
```

```
## Median: 4.158883
```

```
cat("Mode:", mode_value, "\n")
```

```
## Mode: 0
```

```
cat("Variance:", variance_value, "\n")
```

```
## Variance: 36
```

Maximum likelihood estimator (MLE) of the parameters from the Weibull distribution

Probability Density Function ($f(x;\lambda,k)$) for the Weibull distribution:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The likelihood function ($L_{\hat{x}}(\lambda,k)$):

$$\begin{aligned} L_{\hat{x}}(\lambda,k) &= \prod_{i=1}^{n} \frac{k}{\lambda}\left(\frac{x_i}{\lambda}\right)^{k-1} e^{-\left(\frac{x_i}{\lambda}\right)^k} \\ &= \left(\frac{k}{\lambda}\right)^n \left(\frac{1}{\lambda^{k-1}}\right)^n \left(\prod_{i=1}^{n} x_i^{k-1}\right)\left(e^{-\sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k}\right) \\ &= \frac{k^n}{\lambda^{nk}} e^{-\sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k} \prod_{i=1}^{n} x_i^{k-1} \end{aligned}$$

The log-likelihood function ($\ln L_{\hat{x}}(\lambda,k)$):

$$\ln L_{\hat{x}}(\lambda,k) = n\ln(k) - nk\ln(\lambda) - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k + (k-1)\sum_{i=1}^{n}\ln x_i$$

Partial derivative of the log-likelihood function with respect to λ:

$$\frac{\partial \ln L_{\hat{x}}(\lambda,k)}{\partial \lambda} = -\frac{nk}{\lambda} + k\sum_{i=1}^{n}\frac{x_i^k}{\lambda^{k+1}}$$

Solving for the desired parameter of λ by setting the partial derivative equal to zero:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial \lambda} &= 0 \\
-\frac{nk}{\lambda} + k\sum_{i=1}^{n} \frac{x_i^k}{\lambda^{k+1}} &= 0 \\
-\frac{nk}{\lambda} + \frac{k}{\lambda}\sum_{i=1}^{n} \left(\frac{x_i}{\lambda}\right)^k &= 0 \\
-n + \sum_{i=1}^{n} \frac{x_i^k}{\lambda^k} &= 0 \\
\frac{1}{\lambda^k}\sum_{i=1}^{n} x_i^k &= n \\
\frac{1}{n}\sum_{i=1}^{n} x_i^k &= \lambda^k
\end{aligned}$$

Therefore the estimator $\hat{\lambda}$ is:

$$\hat{\lambda} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}$$

Plugging in $\hat{\lambda}$ into the log-likelihood function ($\ln L_{\hat{x}}(\lambda, k)$) and then differentiating with respect to k in order to find the estimator $\hat{k}$:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial k} &= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\lambda - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\left[\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}\right] - \frac{\sum\limits_{i=1}^{n} x_i^k}{\left[\left(\frac{1}{n}\sum\limits_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}\right]^k} + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - \frac{nk}{k}\ln\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right) - \frac{\sum\limits_{i=1}^{n} x_i^k}{\left(\frac{1}{n}\sum\limits_{i=1}^{n} x_i^k\right)} + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - n\ln\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right) - n + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{n}{k} - \left(\frac{n\sum\limits_{i=1}^{n} x_i^k\ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) + \sum_{i=1}^{n}\ln x_i \\
&= \frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n} x_i^k\ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) + \frac{1}{n}\sum_{i=1}^{n}\ln x_i
\end{aligned}$$

Solving for the desired parameter of k by setting the partial derivative equal to zero:

$$\begin{aligned}\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial k} &= 0\\ \frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) + \frac{1}{n}\sum_{i=1}^{n} \ln x_i &= 0\\ \left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) - \frac{1}{n}\sum_{i=1}^{n} \ln x_i &= \frac{1}{k}\end{aligned}$$

Therefore the estimator $\hat{k}$ is:

$$\hat{k} = \left[\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} - \frac{1}{n}\sum_{i=1}^{n} \ln x_i\right]^{-1}$$